DOCOMO Cloud | Android | Application

Technology Reports

# Android Application Development in DOCOMO Cloud

*At NTT DOCOMO, we aim to provide our users with unique, innovative services that combine smartphones with the DOCOMO Cloud. We recently developed Android™*1 smartphone applications in order to implement four cloud services.*

Communication Device Development Department
*Chisa Takeda*
*Reiji Watanabe*
*Yoshitaka Tanemura*
*Shunsuke Fukuda*
Service & Solution Development Department
*Fumihiko Kato*

## 1. Introduction

The spread and diversification of cloud services is also extending to and becoming established in the mobile field. Over The Top (OTT)*2 providers including global players are in the process of shifting the focus of their own cloud services from PCs to mobile devices. This is expected to lead to the development of cloud services that exploit the characteristics of mobile devices, such as their portability and the availability of positional information.

In order to provide our users with unique, innovative services that combine smartphones with the DOCOMO Cloud, we have developed the following Android applications.

- DOCOMO phone book: By linking a cloud-based phone book with a Social Networking Service (SNS), this provides mobile phone carriers with an SNS application and connected notification functions.
- Expanded Shabette-Concier functions: Provides Shabette-Chara functions in combination with Machi-Chara*3. In the DOCOMO network cloud, we are using speech recognition, intention interpretation and speech synthesis.
- Photo collection: Automatically uploads photos and movies from a smartphone. Synchronizes multiple devices.
- Expanded dmarket video viewing functions: Supports multi-device viewing of dmarket content.

This article presents an overview of these four applications.

## 2. DOCOMO Phone Book

In addition to basic phone book functions, the DOCOMO phone book application also provides functions for SNS linkage, and synchronization of phone books in the cloud. The service image is shown in **Figure 1**.

1) SNS Linkage Function

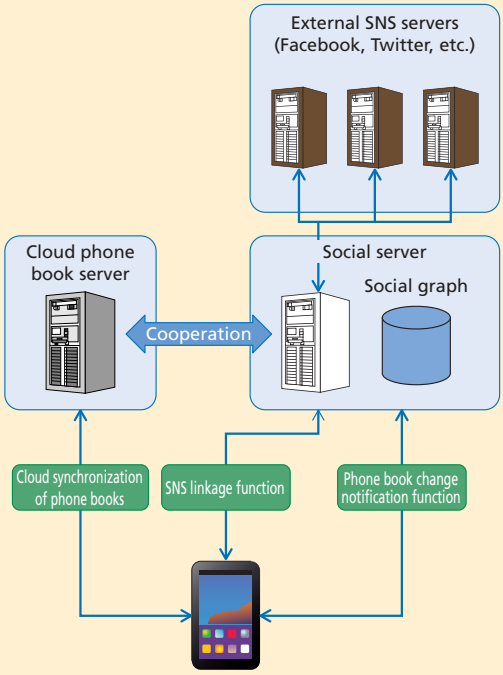
The SNS linkage function has been installed in mobile terminals since the summer 2012 models. It uses a social server to acquire feeds from various social network services, and can display the results on the application's timeline page. It also maintains links with SNS friends and with contacts registered in the phone book, allowing the user to display a timeline for a specific friend or the SNS feeds for names registered in the phone book.

2) Cloud Synchronization of Phone Books

The function for cloud synchronization of phone books provides a means of synchronizing contact data and My Profile*4 data with a server. This makes it easy to migrate data to a new device or recover data when a terminal has



*1 **Android™**: A trademark or registered trademark of Google, Inc., in the United States.




Facebook: A trademark or registered trademark of Facebook, Inc.
Twitter: A registered trademark of Twitter, Inc. in the USA and other countries.

**Figure 1 Phone book service concept**

been lost. The social server can also draw a social graphs[*5] by calculating relationships between the users of the cloud phone book synchronization function based on the contact data and My Profile data they have uploaded. In the DOCOMO phone book service, based on this social graph, we provide a phone book alteration notification service that notifies a user's friends when changes are made in My Profile, and a Friend News function that allows a user to browse posts submitted by friends via the DOCOMO phone book social graph.

In the DOCOMO phone book application, it is possible to synchronize up to 3,000 items or 50 MB of contact data with 200 groups and the user's profile information in the cloud. With the contact data, it can also synchronize business card data, the icon/color settings of groups and information relating to the SNS collaboration feature of the DOCOMO phone book.

During cloud synchronization, it is possible to transmit and receive up to 50 items of contact data in a single transaction. By processing multiple items of data at a time, we aim to reduce the communication time and the time needed to store this data in the terminal's DB.

Since images and business card data consist of relatively large amounts of data in the phone book, they are not uploaded or downloaded when synchronizing related contact data unless the data itself has changed. In this way, the amount of data communication needed for synchronization is reduced.

3) Phone Book Alternation Notification Function

We implemented the following framework for reporting changes in the phone book (**Figure 2**).

(1) When a user changes his or her profile in the DOCOMO phone book application, it initiates synchronization with the cloud phone book server.

(2) A friend notification setting page is displayed based on the contacts uploaded to the cloud phone book server, and the user can select which friends the profile information should be reported to.

(3) When this selection is made, the social server reports this information on the terminals of the selected friends.

(4) On the terminals that receive this notification, the DOCOMO phone book application fetches the updated profile information from the cloud phone book server.

(5) This information is shown to the user within the application, and based on the user selection, the change of the data is reported to the cloud phone book server.

---

*2 **OTT**: The provision of services such as VoIP and instant messaging as a layer on top of the communication carrier's network. These services are not subject to geographical or carrier restrictions.

*3 **Machi-Chara**: A service that displays a character that is chosen by the user on the standby screen or menu screen of a mobile terminal.

*4 **My Profile**: A function that enables the user to register his/her phone number on a mobile terminal together with other personal data such as the user's e-mail address and postal address.

*5 **Social graph**: Data that expresses relationships by connecting people with lines. In the cloud phone book service, a social graph is constructed from phone book data uploaded to the cloud.

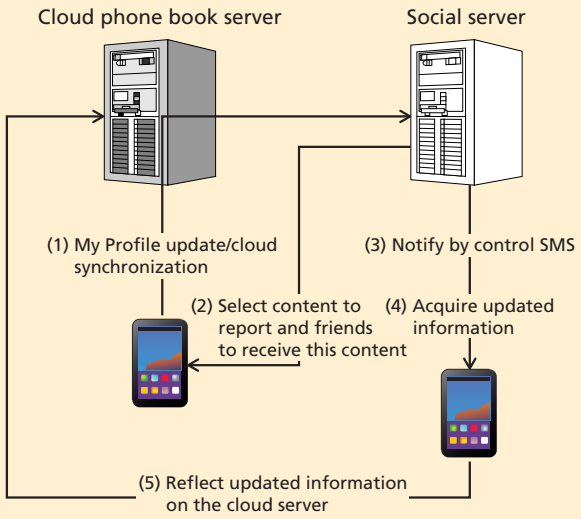



**Figure 2 Phone book change notification mechanism**

The phone book update notification function can notify friends of changes in a user's profile information by e-mail or other such means, and simply by selecting the updated information at the receiving side, it is possible to update the phone book to the most recent status.

4) Friend News Function

When a user has registered SNS information in a phone book, the user can select in the friend notification settings screen whether or not to publish SNS information. If the user chooses to do so, the phone book social graph constructed by the social server can be used to deliver SNS tweets and posts regardless of the relationship settings of each SNS. In this way, users can browse the tweets and posts of friends registered in the phone book even if they are not members of the same SNS, thereby promoting communication between users.

## 3. Expanded Shabette-Concier Function

Shabette-Concier is a service that we launched on Android smartphones in March 2012, whereby a smartphone can immediately respond to spoken commands to display information or run a particular function.

1) Processing Flow

The flow of processing that occurs when using this service is shown in **Figure 3**.

(1) The user starts the Shabette-Concier application, and speaks the name of the desired information or service.
(2) The application transmits the speech data input by the user to a cloud speech recognition server.
(3) In the cloud, the speech is analyzed and converted into text, which is transmitted back to the application.
(4) The application transmits the speech recognition results to an intention recognition server in the cloud.
(5) The user's intention is analyzed in the cloud to produce an operating instruction, which is converted into a command and transmitted to the application.
(6) Based on the command received at step (5), the application displays a character response message and performs corresponding actions such as showing the information search results, running a particular function or reading out a response. When reading out a response message, the text received for this purpose at step (5) is transmitted to a cloud speech synthesis server.
(7) The speech is synthesized in the cloud, and the speech data is transmitted back to the application.
(8) The application plays back the speech data received at step (7).

The speech recognition, natural language processing and speech synthesis processes used by this service must be updated from time to time so that they can handle new words and languages, and providing them on the cloud makes it possible to add and expand their functions in a flexible way.

2) Command Control Framework

When an application transmits the user's speech to an intention recognition server in the cloud, it also sends a

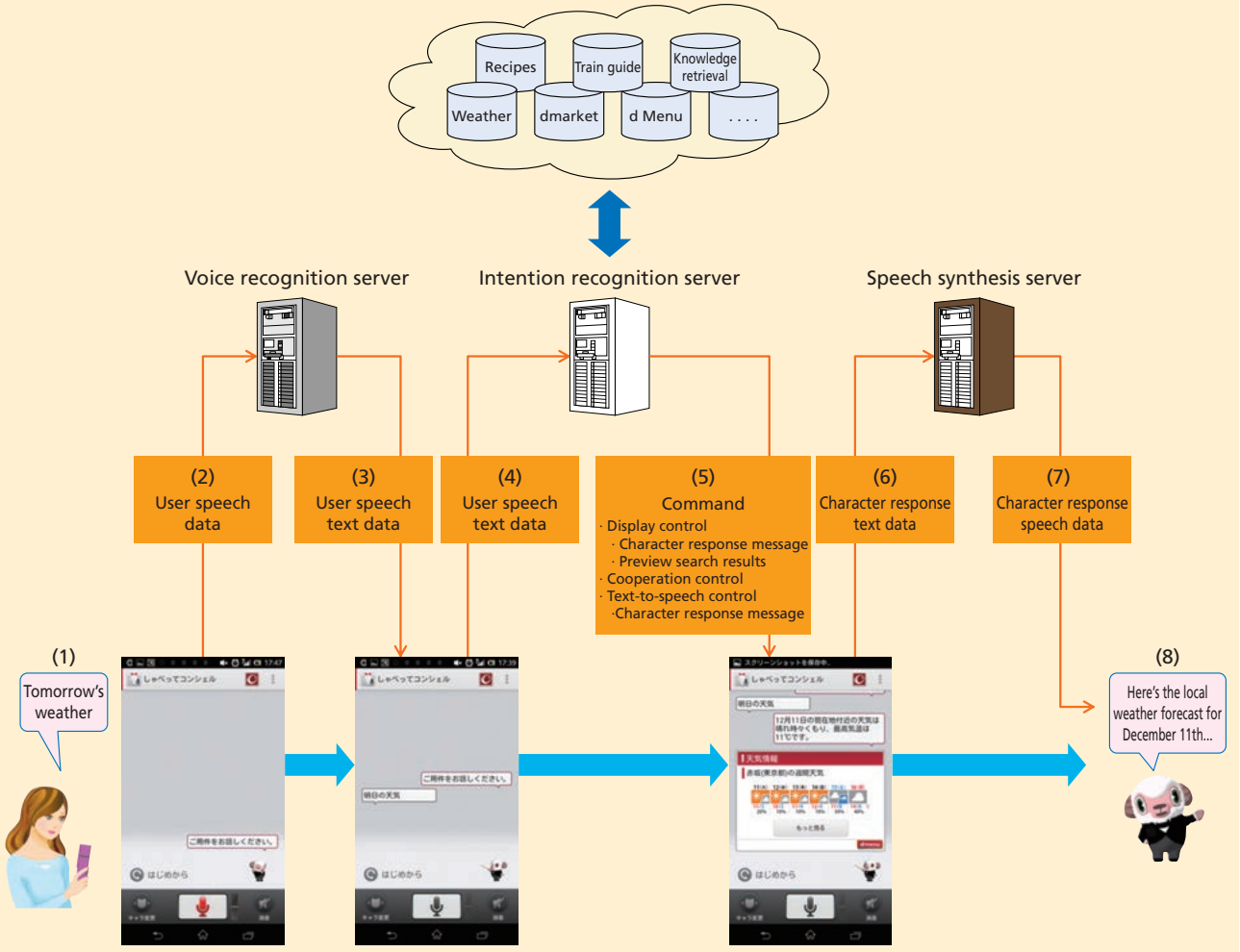



**Figure 3 Flowchart of the Shabette-Concier service**

report containing other information including terminal-related details such as the user's current location and device details, and the results of the previous operation. The intention analysis server generates commands by analyzing the user's intentions based on this information.

The commands mainly consist of three types: display control (e.g., previewing data search results, showing "Yes"/"No" buttons or the like or displaying Shabette-Chara actions) cooperation control (cooperating with other applications) and reading control (reading the Shabette-Chara responses and the results of data searches). The intention recognition server transmits multiple commands of this sort by combining them with the order in which they are to be performed by the application. When the application receives these commands, it runs the specified commands in order to provide a response to the user.

To allow commands to specify conditional branches, it is possible to provide different responses depending on the availability or quantity of information search results, or the availability of an application that supports cooperation.

3) Extension Functions

(1) Response control for individual characters in Shabette-Chara

There are three factors that determine the characteristics of Shabette-Chara: the messages it uses to respond to user utterances, the actions it performs and the voice it uses. For each factor, it is possible to control the response of each character by providing a database in the cloud.

For general conversation, the response messages and actions are associated with each character in the intention recognition server so that the responses can be adapted to the character's characteristics.

Also, each character's voice is generated by registering pre-recorded speech as data for synthesis in a speech synthesis server. New characters can be added by adding this synthesis data to the speech synthesis server.

(2) Expanded call functions

In Shabette-Concier, it is possible for cooperation to take place between a variety of applications and functions inside a terminal, but there has been an issue in that the application itself must be updated when cooperating with new applications or new interfaces. We have now resolved this issue by using the cloud to store the actual information necessary for cooperation with other applications (including package names, class names and other extended information to be handed over to the applications). On the application side, information received from the cloud is used to perform cooperation by dynamically generating cooperation information.

This development makes it possible to rely on the cloud to adapt to new forms of cooperation with terminal applications, and to control how cooperation is implemented on devices with different cooperation capabilities, allowing for a flexible and rapid response.

## 4. Photo Collection

Photo Collection is a service that promotes the use of photos by uploading photos and movies (hereinafter referred to as photos) from a terminal to a server, and organizing them into groups corresponding to different people or events, for example. This service can collect photos not only from terminals but also from Web browsers, PC applications and other services (including Eye-Fi®*6). The photos collected together on a server can also be browsed and organized in real time on multiple devices.

The process flow of the Photo Collection service is shown in **Figure 4**, and the functions of each service are described from the terminal's point of view.

1) Synchronization Function

The Photo Collection synchronization function is a function that reflects changes mutually between the server and terminals in real time when adding or modifying photos or the group information (described in detail below) applied to photos, either on a terminal or on the server. When this application detects that new photos have been added in the terminal, either by taking photographs or by downloading pictures, it automatically uploads them to the server. The server generates thumbnail images and resized images from the uploaded photos, and generates the abovementioned group information. The terminal downloads these thumbnail images and group information from the server. In this way, photos that have been newly collected on a server from other terminals can be synchronized across multiple devices by downloading thumbnail images or the like of the added photos from the server. The specific details of the synchronization function are described below.

First, it is presumed that photos that have been uploaded from other terminals must be synchronized with the local terminal. However, even if a user has synchronized all the photos, this user will not necessarily view all of these photos. Therefore, the terminal only synchronizes the group information and thumbnail images (which have smaller file sizes) from the server. When the following processes have

*6 **Eye-Fi®**: A registered trademark of Eye-Fi Japan, Inc.

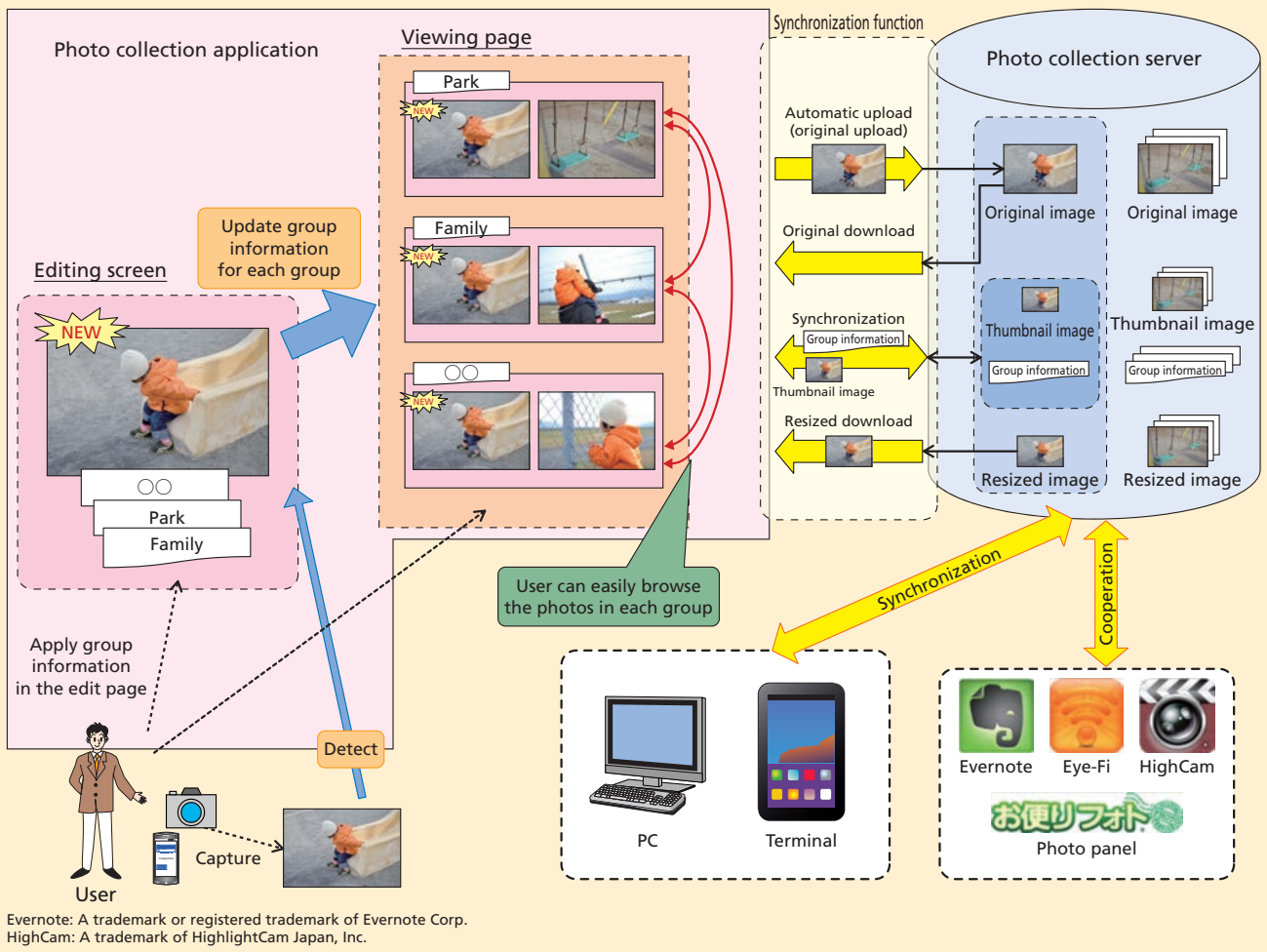



**Figure 4 Flowchart of Photo Collection service**

occurred, the terminal receives a change notification from the server and performs the acquisition process.

- Photos uploaded from another terminal or service
- Group editing request received from the terminal

On the other hand, the resized images are acquired by sending a request to the server on the first occasion when the user displays the photo at full-screen dimensions within the application.

In this way, we can minimize the amount of traffic while implementing a synchronization function that keeps track of the latest updates.

2) Group Editing Function

This function is used to organize photos into categories corresponding to different people or events, for example. The user can assign an individual photo to more than one group, and when groups are edited on a terminal, the results are transmitted to the server as group information associated with the photos. The flow of group editing between the terminal and server is described below.

When a new photo has been uploaded to the server, the server sorts it automatically to form an untitled group, which it synchronizes with the terminal. The terminal performs group editing on the photos in the untitled

group and any other groups. With this process, the group information is set according to the user's wishes, and is transmitted to the server, whereby these changes are reflected on the server. The terminal that performed group editing not only reports these editing results to the server, but also reflects them inside the terminal at the same time. This process makes it possible to check the editing results on the screen in real time while minimizing the amount of traffic exchanged with the server and avoiding the need to wait for the results to be reflected in the server.

3) UI

One of the concepts behind the creation of the UI in this service was to enable users to rediscover images from the past. To realize this concept, we implemented the UI based on the following features.

(1) Random display on top page

(2) Page transitions specialized for group displays

In (1), the user's opportunities for rediscovery are increased by displaying images every time the application is started up. The user can also tap these images to display them in full-screen mode, so there is also a line connecting to (2). In (2), by providing a UI with a flat structure based on group information instead of an ordinary hierarchical UI, we can provide more lines for direct transition between groups. Using this UI's transition configuration, it is possible to display not just an individual photo, but also a series of photos that are connected with this photo. As a result, the user can discover newer photos by repeatedly transitioning from one photo to the next.

As an overall implementation of this UI, the synchronization function 1) displays photos in the terminal without accessing the server each time the user moves to a different page. As a result, the processing time of the UI display is reduced.

## 5. Expansion of dmarket Application Video Viewing Function

In the dmarket application, cloud services are implemented in conjunction with a media player application.

Specifically, we have introduced a framework whereby a user who owns multiple terminals can play back content on one terminal and then continue playing back the same content on another terminal. The dmarket application provides two ways of viewing content — streaming[*7] and download. In each case, multi-device functions are implemented as shown in **Figure 5**.

1) Viewing Streaming Content

When viewing streaming content, the user selects the content to view (viewing start request), whereupon the dmarket application is able to receive not only the corresponding content URL but also a playback location (time information showing the point from which to start playing back the content). The dmarket application starts up the media player application by passing this information as a parameter to initiate playback.

When the user stops playing back the content or quits the media player application, the media player transmits the playback position to the server. When this sequence of operations is performed on multiple terminals belonging to the same user, the user can continue to play back content from the same location on a different terminal.

In addition to this process, the application also supports transmitting the playback location periodically while playing back content in order to deal with cases where it becomes impossible to transmit the playback location, such as when the user moves out of communication range while content is playing.

2) Viewing Downloaded Content

An issue that arises when playing back downloaded content is that unlike the playback of streaming content, there may be no network connection when the content playback is started. We must therefore consider situations where it is not possible to acquire the start location from the server.

We therefore adopted a method whereby the playback location is also stored inside the terminal, which either uses this information or the information received from the server, depending on the availability of a network connec-

*7 **Streaming**: A communication method for sending and receiving audio and video data over the network, whereby data is received and played back simultaneously.

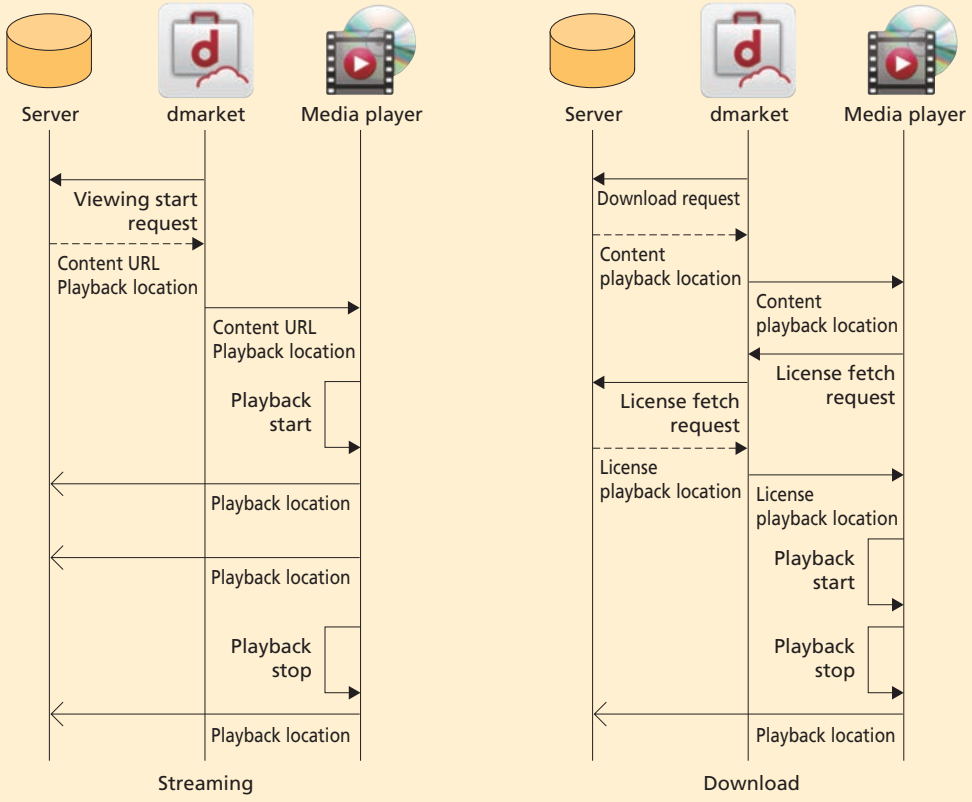



**Figure 5 dmarket multi-device functions**

tion. Specifically, it records the time at which the playback location is stored, and the playback start time is determined by comparing the information received from the server and the information inside the terminal, and choosing whichever is more recent.

When an attempt is made to acquire the playback position from the server every time a content playback operation is performed, the content playback cannot start until a response has been received from the server. If the network is busy or a connection cannot be made, then there may be a considerable delay before playback can begin.

We therefore used a framework that expects to acquire the playback location in situations such as when downloading content or acquiring a license, and does not acquire it when the content is played back.

Furthermore, as with streaming playback, it may be impossible to transmit the timing at which playback is stopped. However, when playing back downloaded content, it is considered undesirable to perform periodic communication, so we used a framework where the playback position is transmitted only when the playback is stopped. Also, to deal with cases where communication is not possible, we added a function that transmits all the playback positions inside the terminal to the media player at once.

In this way, we implemented sharing of playback positions across multiple terminals for both streaming and downloaded content. Furthermore, by using the same content IDs to manage the playback positions of streaming content and downloaded content, we have also implemented a function that, for example, allows a user to watch downloaded content on terminal A and then continue watching the same content by streaming playback

on terminal B.

## 6. Conclusion

In the future, we hope to provide multi-device compatibility for the phone book cloud service, and a function that publishes photos organized into photo collections to other users (including users of other telecommunication carriers). We also hope to provide print production services in conjunction with online printing businesses.

We also plan to expand the various functions and content to meet users' needs flexibly by exploiting cloud mechanisms in other applications.